MA 3046 - Matrix Algebra
Objectives

I. Introduction

This course provides students in the engineering and physical sciences curricula with an applications-oriented coverage of the major topics of matrix and linear algebra. Principal results will be examined not only from theoretical and geometric aspects, but also from the perspective of how those results can actually be implemented numerically, and the relative complexity and expected accuracy of that implementation.

II. Objectives

Upon successful completion of this course, one should be able to:

1. Define the standard inner product in $\mathcal{C}^n$. Calculate standard inner products in $\mathcal{C}^n$. State the definitions of orthogonal and orthonormal vectors and sets in $\mathcal{C}^n$. Describe the relationships between orthonormal bases, the general coordinates of a vector in terms of such a basis, and systems of linear equations.

2. State the definitions of the Hermitian (adjoint) of a matrix, and of Hermitian and skew-Hermitian matrices. Describe how the these concepts simplify for the case of real matrices.

3. State the definition of a Unitary matrix, and describe the relationship between unitary matrices and orthonormal vectors.

4. Describe the general properties of any vector norm. Given an appropriate vector, find $\| \mathbf{x} \|_1$, $\| \mathbf{x} \|_2$ and $\| \mathbf{x} \|_\infty$. Describe how matrix norms are derived from corresponding vector norms. Show that multiplication by unitary matrices preserves both Euclidean vector and matrix norms, and the angles between vectors.

5. State the definition of the singular values of a matrix, and of the singular value decomposition (SVD) of a matrix. Find the SVD for small, simple matrices. Describe the relationships between the singular values of a matrix, the rank of that matrix, and its Euclidean and Froebenius matrix norms. Describe when and how the SVD can be used to construct low-rank approximations to a given matrix.

6. Describe the general characteristics of any finite-length, non-decimal, normalized floating point number system, such as those commonly used to represent real numbers in computers, and the primary errors and difficulties that can arise in matrix computations because of such representations.

7. State the definition of a floating point operation (flop), and discuss the general factors that influence the speed at which various computers may execute different flops. State the approximate number of flops required for scalar multiplication, matrix addition, matrix-vector multiplication and matrix-matrix multiplication. Determine approximately how many flops will be required by a given matrix or vector computation, and whether and, if so, how that computation may be rewritten into a more flop-efficient form.

8. Describe, in general, how both computer hardware and software architecture can significantly impact the actual efficiency of matrix calculations. Specifically, define and address the effects of serial versus parallel versus pipeline (vector) architectures, and of cache, RAM and swap. Distinguish between complied and interpreted languages, and row versus column-oriented storage, and their expected effects on matrix computations. Describe the general concept behind and expected utility of routines such as the BLAS.

9. Program specific matrix and vector calculations in MATLAB so as to both approximately minimize the number of flops required, and to as utilize, efficiently as possible, other MATLAB and hardware aspects.

10. State the definition of a projector and of an orthogonal projection, and describe the relationships between projectors and complementary subspaces. Interpret both non-orthogonal and orthogonal projectors geometrically in $\Re^2$ and $\Re^3$. Describe how general projectors onto subspaces in $\mathcal{C}^n$ may be computed given appropriate bases, and how these computations simplify for orthogonal projection, both with and without orthonormal bases.

11. Use both the classic and modified Gram-Schmidt procedures to reduce an appropriate linearly independent set to an orthonormal one, and to compute the **QR** factorization of a given matrix. Describe why the modified procedure is generally superior.

12. Compute alternative **QR** factorizations using both Householder reflections and Givens rotations, and describe the advantages and disadvantages of these approaches versus Gram-Schmidt.

13. State the approximate number of flops required to produce the **QR** factorization of a given matrix by the Gram-Schmidt, Givens and Householder methods.

14. Describe the relationships between orthogonal projection, least-squares and Fourier methods. Describe the basic concepts behind the normal equations for least squares, and the limitations of the method. Apply the normal equations to solve a given overdetermined system of linear equations. Describe the advantages of **QR** or the SVD factorization over the normal equations for least squares problems.

15. Describe the general meaning of the terms stability and condition as they apply to numerical computations. Describe, in general terms, the difference between an ill-conditioned problem and an unstable algorithm. Identify which commonly-used algorithms are unstable, conditionally stable and unconditionally stable.

16. State the approximate number of flops required in Gaussian elimination to reduce a matrix to upper triangular form, and to solve either an upper or lower triangular system of equations.

17. Perform Gaussian elimination with a stated pivoting or scaling strategy to solve a given system of linear equations, including correct simulation of the effects of floating-point arithmetic.

18. Describe the **PA**=**LU** and Cholesky factorizations of a given matrix $\mathbf{A}$, including the differences between them and how they are related to Gaussian elimination and to elementary matrices. Given an appropriate matrix, compute any of these factorizations, and use the results to solve the system $\mathbf{Ax} = \mathbf{b}$. Describe the advantages and disadvantages of storing coefficient matrices in such factored forms.

19. Define the terms error, residual and condition number in the context of numerical solution of systems of linear equations, and describe the relationships between them.

20. Describe iterative improvement (iterative refinement) as it applies the solution of systems of linear equations, including the principle and assumptions underlying it, and what one can learn from it. Simulate iterative improvement on a given system of equations using a specified floating-point arithmetic.

21. Describe how the eigenvalues and eigenvectors of Hermitian, Unitary and Normal matrices differ from those of general matrices.

22. State the definitions of Quadratic Form, and of Positive and Negative definite and semi-definite. Describe the general properties of the eigenvalues and eigenvectors of a quadratic forms, and how the eigenvalues relate to positive or negative definiteness.

23. Use basic, inverse or shifted inverse power method iteration to either find the largest eigenvalue of a matrix or to refine estimates of other eigenvalues. Describe when and why these methods should converge.

24. Describe how Householder reflections can be used to reduce a given matrix to upper Hessenberg form. Apply them in appropriate cases. Describe how the **QR** algorithm can be used to find the eigenvalues and eigenvectors of an upper Hessenberg matrix.

25. State the relation between eigenvalues, eigenvectors of a matrix and its SVD. Given an appropriate matrix, use eigenvalue and eigenvector methods to find its SVD.

26. Describe and use the Jacobi, Gauss-Seidel and Successive Overrelaxation (SOR) iterative methods for solving $\mathbf{Ax} = \mathbf{b}$; explain when such methods are used; give sufficient conditions for convergence; and tell why SOR is usually superior to Gauss-Seidel, which in turn is usually superior to Jacobi.

27. Given library routines for the solution of the above problems as well as other utility routines, be able to indicate how to use them in the solution of a larger problem involving the solution of several 'subproblems'. For example

    Assume that $\mathbf{A}$ is a large matrix. Using appropriate MATLAB routines, write a program to find the smallest possible matrices $\mathbf{U}$, $\mathbf{V}$ and Σ, such that

    $$\frac{\| \mathbf{A} - \mathbf{U}\Sigma\mathbf{V}^T \|}{\| \mathbf{A} \|} < .01$$

    Apply this program to a random 100×100 matrix, and analyze its performance in terms of both numerical efficiency and accuracy.

III. Assignments, Examinations and Grading

There will be one in-class hour exam (midterm) plus a comprehensive two-hour final. All of these exams will be closed book, but one page (8 1/2 x 11) of notes (one side only) may be brought to the midterm, and two pages to the final. The midterm will count for approximately 15% of the total course grade, and the final for approximately 35%.

There will also be a number of in-class laboratories, designed to illustrate and emphasize certain aspects of both software and algorithms. Many of these will be ungraded, but some will require the student complete and turn in a scripted worksheet, which asks the student to complete several rudimentary exercises, and compare the results to appropriate theory. Most of these should be able to be completed during the scheduled laboratory time, and most answers on them can be hand-written and should require not more than a sentence or two. The combined grades on these laboratories will determine approximately 15% of the course grade.

Finally, there will also be a number of graded projects. These will require completing a report in which the student will be asked to write and critically analyze either their own implementation of a appropriate algorithm or a program use existing, state-of-the-art software to solve a reasonably realistic problem in engineering and scientific computation. The combined grades on these projects will determine approximately 35% of the course grade.

IV. Guidelines for Laboratory and Programming Assignments

All graded programming assignments should be essentially individual efforts. This does **not** mean that students may not talk over with someone else how to attack the problem, or consult them during check-out in an effort to find bugs - in fact both of these are *highly* encouraged. However, each student must write the actual program code and documentation themselves, since the grades on these assignments will be a significant component of the course grades. Furthermore, whether students discuss the project with

others or not, they should not turn in an assignment until the are personally comfortable explaining exactly what they did and why.

Students should not hesitate to consult with the instructor regarding homework assignments, programming practices, or when they can't locate some elusive "bug." The purpose of these assignments is to illustrate one or more features of the course work. Thus, I do not expected that students will need to spend all of one or more nights trying to get a simple program to run. (A good rule of thumb is 15-20 minutes - i.e. if a student has spent that much time "spinning their wheels" and has no idea what is wrong with a program or problem, etc., ask for help.) Lastly, since the programs are often designed to illustrate various numerical phenomena, students should be especially careful to check their output before they submit the project to see if they *believe* the answers they've computed.

V. Conclusion

The analysis of numerical methods and algorithms for matrices is important because virtually all of the students in this course will use the computer methods to solve scientific and engineering problems during their studies. For many of them, computational results will constitute a major portion of their thesis. This course not only examines methods, but also provides an opportunity to gain computational experience in the use of high-quality library software through the programming assignments made during the course. These assignments are designed to illustrate the theory and provide experience in obtaining numerical solutions of engineering and scientific problems. In addition, some of the assignments are used to exhibit common computational difficulties and ways of avoiding them.